

Programación científica y análisis estadístico con Python

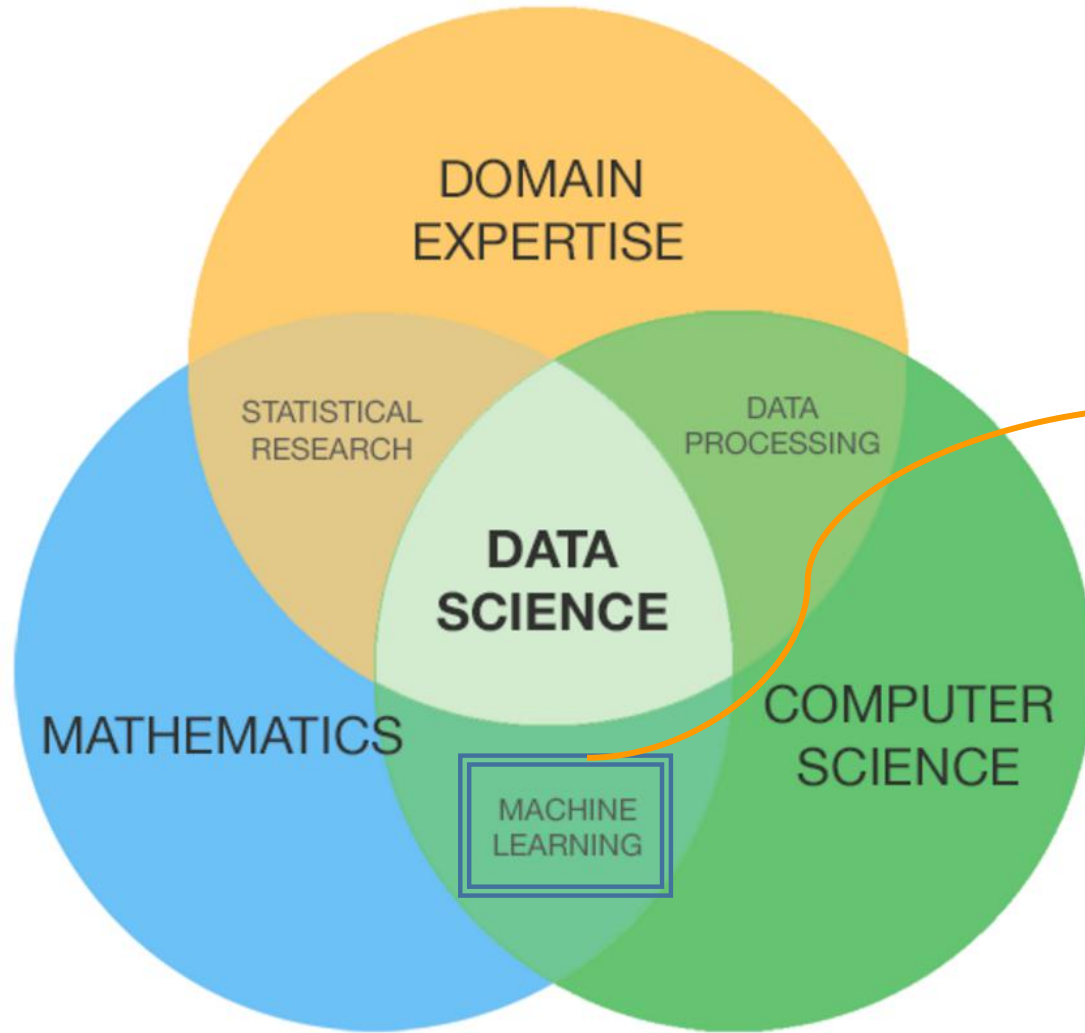
Master en Big Data

Sección 1: Anaconda, Jupyter, elementos básicos, Numpy

PROFESOR/A

Guillermo González Sánchez

Ciencia de datos: Machine Learning ó Statistical Learning



- **Matemáticas, estadística y algoritmia.**
- **Programación**
- **Conocimiento de dominio**

Los lenguajes del análisis estadístico y machine learning



- **Lenguaje de propósito general**
- **Entorno de computación científica avanzado . Emula a Matlab.**
- **Múltiples librerías desarrolladas para ciencia de datos como lo son scikit-learn, scipy, numpy, pandas, tensorflow, keras**
- **Mayor facilidad de computación multiprocessing ó gpu**



- **Lenguaje orientado al análisis estadístico .**
- **La comunidad científica implementa los resultados de sus artículos primero en R.**
- **Abundante colección de pequeñas librerías con multitud de aplicaciones.**
- **Uso directo de los últimos resultados en investigación.**

Entornos de Python científico

os centraremos en el entorno científico de Python orientado a análisis de datos



Las librerías que trae integradas:

- Numpy - cálculo numérico y álgebra lineal
- Pandas - tratamiento de tablas (DataFrame)
- Scipy - funciones científicas/estadísticas
- Statsmodels - modelos estadísticos
- Scikit-learn - algoritmos y preprocesamiento de Machine learning
- Matplotlib - representación gráfica
- IDEs - Jupyter Notebook y Spyder

Instalamos Anaconda en Linux , Mac ó Windows

<https://www.continuum.io/downloads>

la web vienen instrucciones de instalación estrictas para cada sistema operativo

Ejecutamos un Jupyter Notebook/Spyder ejecutando:

- **Linux**: en el terminal escribe `jupyter notebook` ó `spyder`
- **Windows**: el la ventana de comandos (Win+R) escribimos `jupyter notebook` ó `spyder`

3.1.2. Change Jupyter Notebook startup folder (OS X)

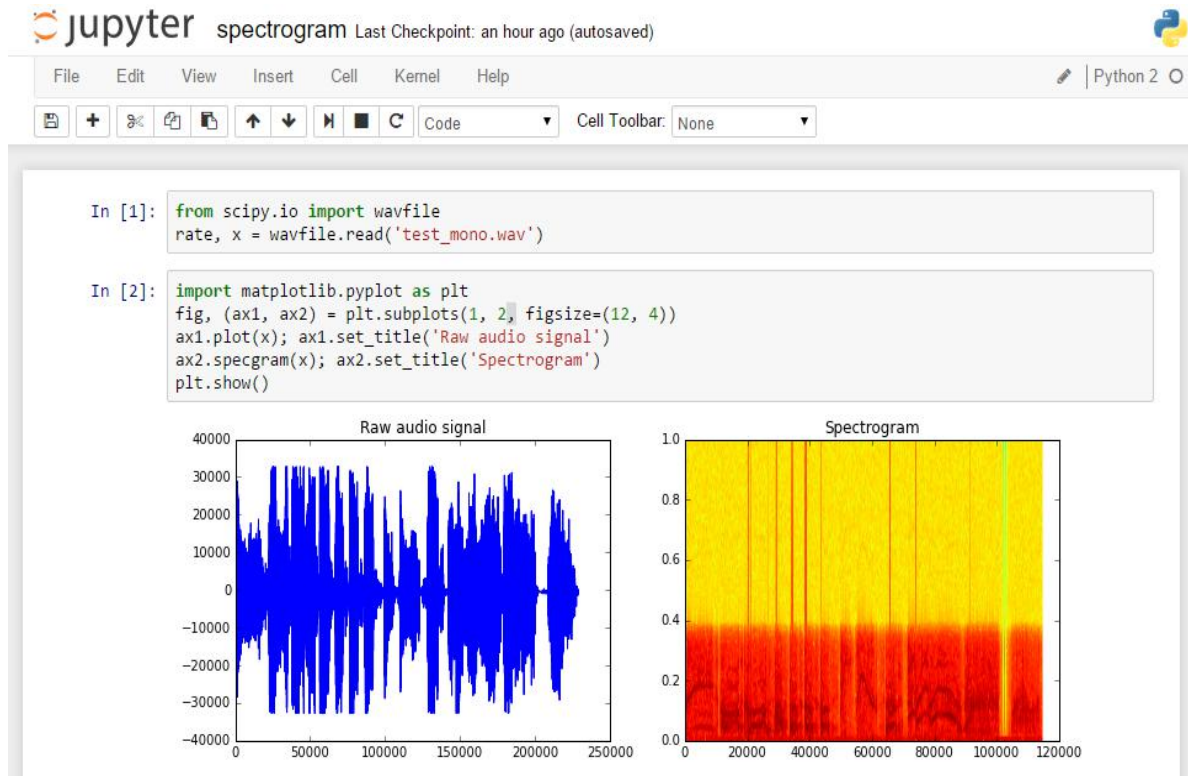
- **MAC**:

To launch Jupyter Notebook App:

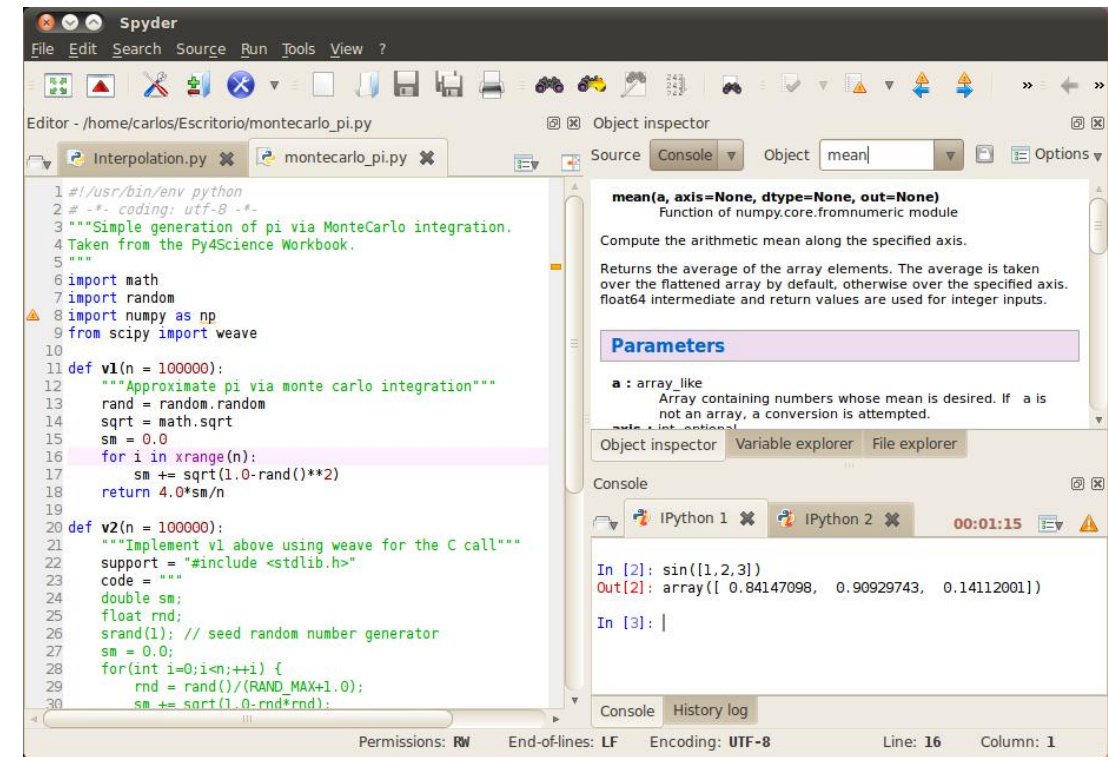
- Click on spotlight, type `terminal` to open a terminal window.
- Enter the startup folder by typing `cd /some_folder_name`.
- Type `jupyter notebook` to launch the Jupyter Notebook App (it will appear in a new browser window or tab).

IDEs para Python orientados a Data Science

Jupyter Notebook



Spyder



Python: listas, tuplas y diccionarios

Las estructuras esenciales de datos en Python son:

- **Listas:** secuencias ordenadas con datos de tipo variable. Se indexan por el orden en el que aparecen. Son mutables y se escriben entre corchetes.

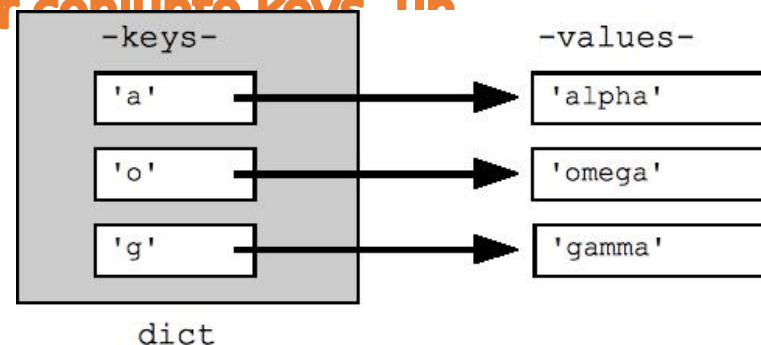
```
l = [2, [3, 5, 6], 'naranja', 6, 'azul']
```

- **Tuplas:** son el mismo objeto que una lista con la diferencia de que no son mutables y se escriben entre paréntesis.

```
tupla=('a',3,'b',7)
```

- **Diccionarios:** es un objeto que representa una aplicación entre dos conjuntos, asociando a cada elemento del primer conjunto keys un elemento en values.

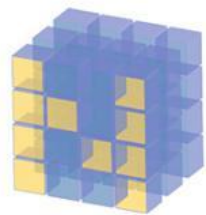
```
{'lea': 'hola', 'lea': 'hola', 12: 90, 'yes': 'no'}
```



Numpy: librería de cálculo numérico y álgebra lineal

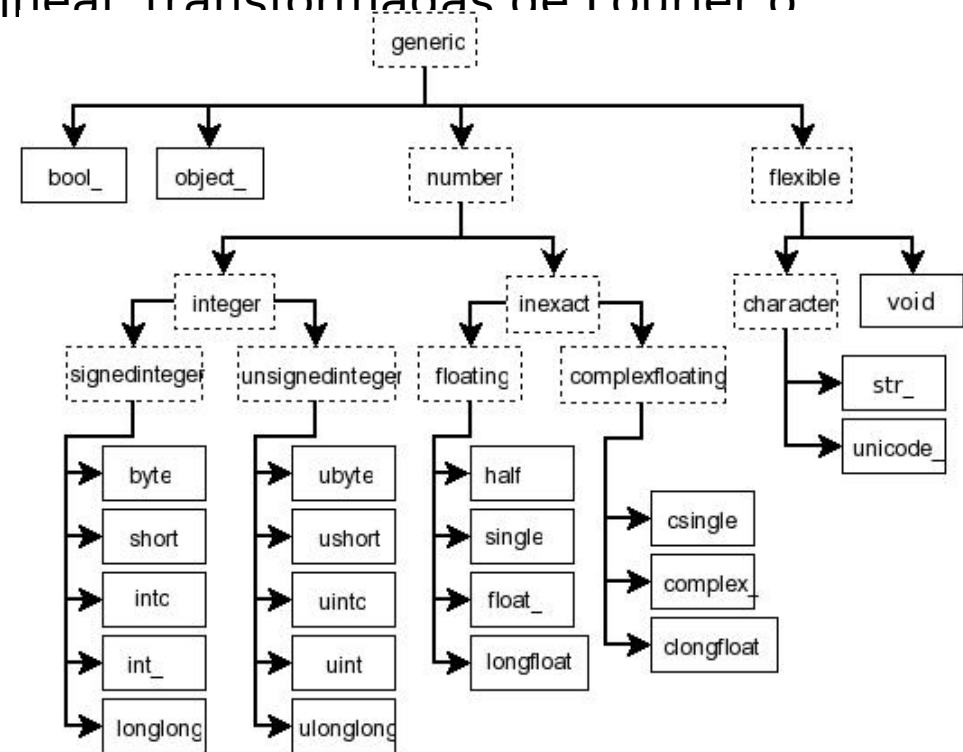
Numpy es un paquete fundamental para hacer computación científica en Python.
Contiene :

- Una potente clase de **arrays N-dimensionales homogéneos**
- Funciones *broadcasting* sofisticadas sobre los arrays
- Funciones matemáticas específicas de álgebra lineal transformadas de Fourier ó generadores de números aleatorios.



NumPy

		axis 1		
		0	1	2
axis 0	0	0,0	0,1	0,2
	1	1,0	1,1	1,2
	2	2,0	2,1	2,2



Numpy: librería de cálculo numérico y álgebra lineal

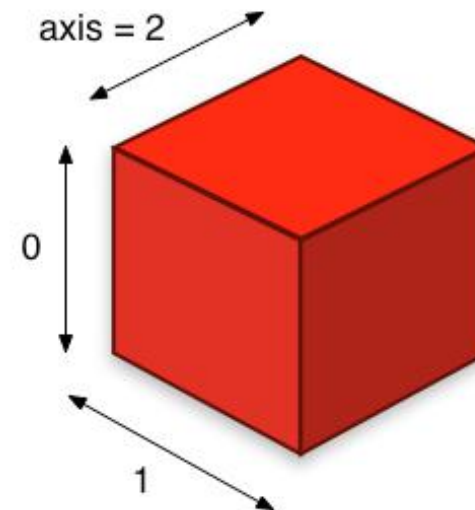
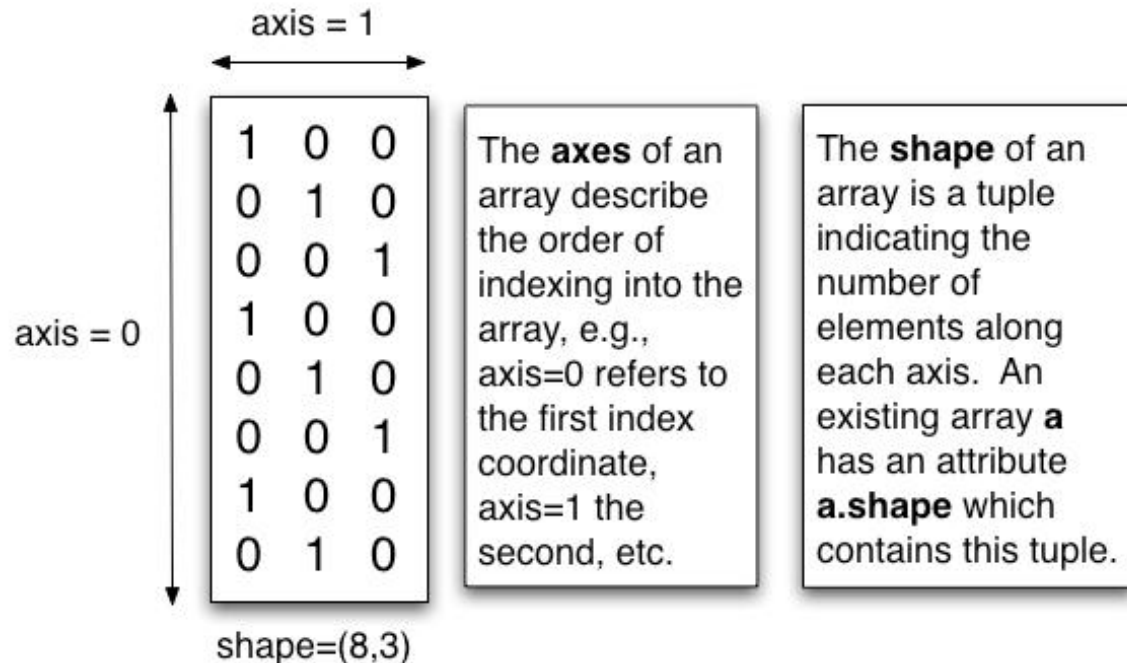
Características esenciales de Numpy

- Operaciones sobre arrays para filtrado, transformación y obtención de subconjuntos.
- Ejecución de algoritmos típicos sobre arrays como ordenación, eliminación de elementos duplicados, etc.
- Operaciones estadísticas.
- Tratamiento de datos para manipulación y unión de conjuntos.
- Definición de expresiones lógicas como sustitución de bucles con if-else.
- Manipulación de datos a nivel de elemento o grupo.

Numpy: librería de cálculo numérico y álgebra lineal

Los **arrays** de numpy son estructuras **n-dimensionales** de datos **homogéneos** organizados

Anatomy of an array



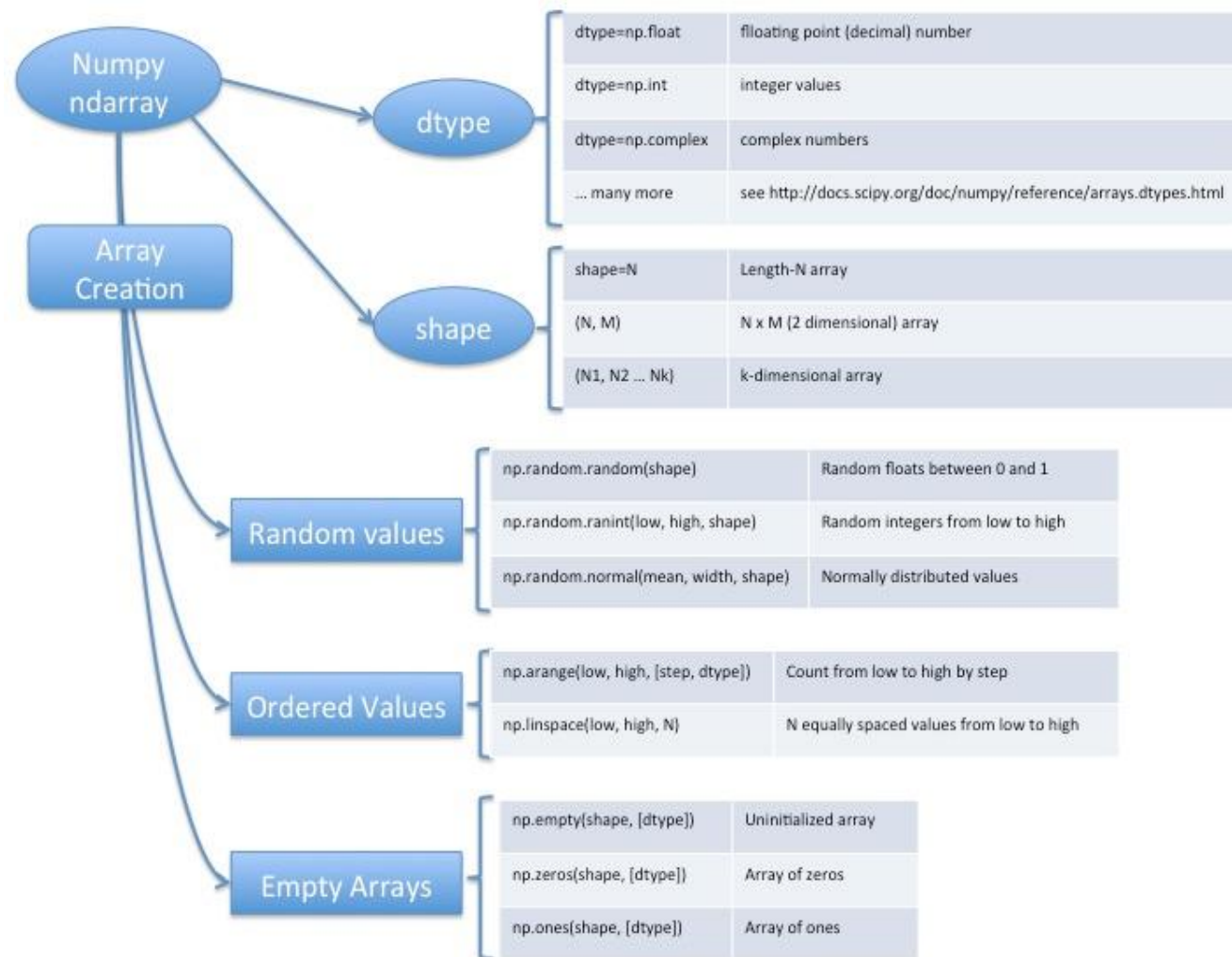
- all elements must be of the same dtype (datatype)
- the default dtype is float
- arrays constructed from list of mixed dtype will be upcast to the "greatest" common type

Numpy: librería de cálculo numérico y álgebra lineal

Tipos de datos

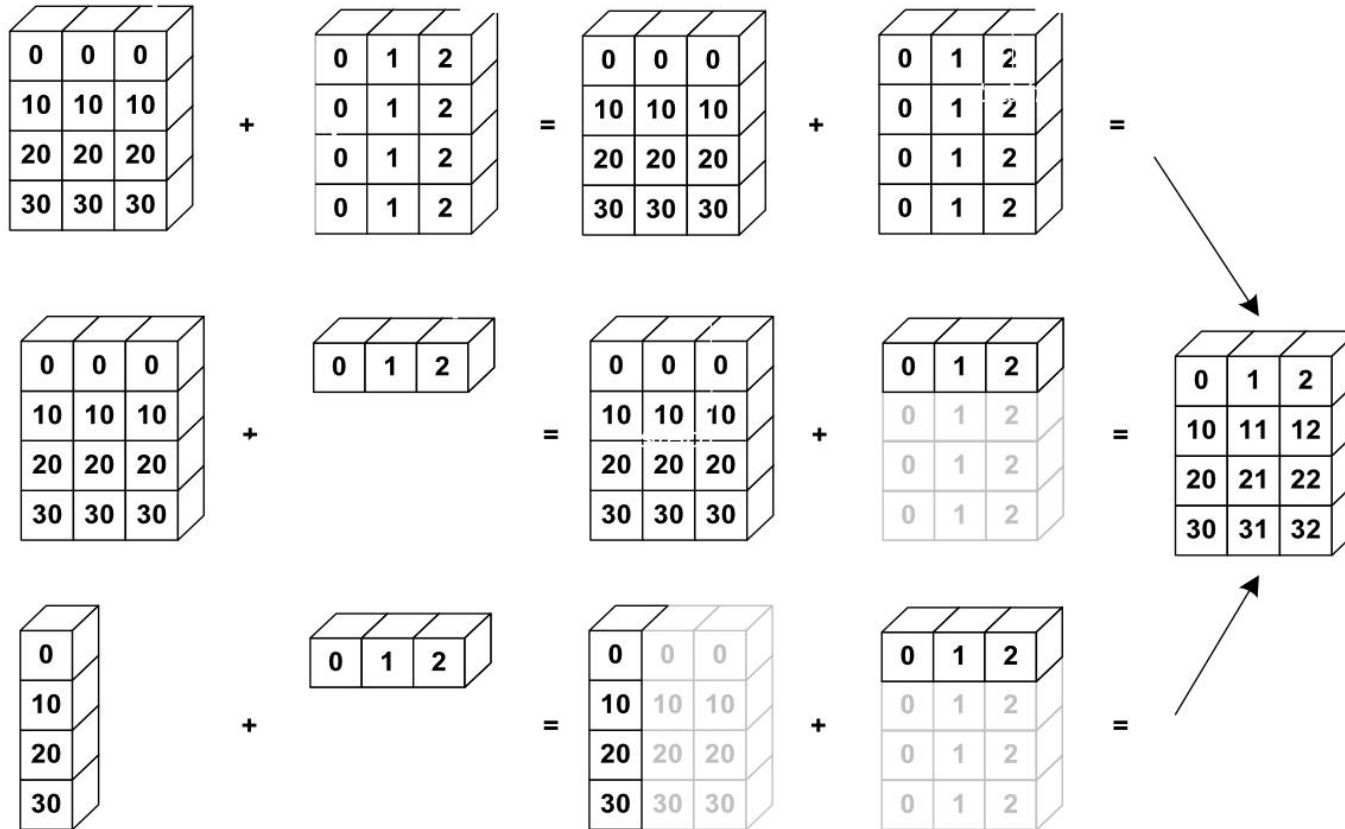
- **int8, uint8**: Tipo integer de 8 bit con signo y sin signo
- **int16, uint16**: Tipo integer de 16 bit con y sin signo
- **int32, uint32**: Tipo integer de 32 bit con y sin signo
- **int64, uint64**: Tipo integer de 64 bit con y sin signo
- **float16**: Tipo float de 16 bit (Half-precision)
- **float32**: Tipo float de 32 bit. Es el tipo float estándar. Compatible con el tipo float del lenguaje C
- **float64, float128**: Tipo float de 64 y 128 bit. Es el tipo float de doble precisión estándar. Compatible con el tipo double del lenguaje C y con el tipo float de Python
- **complex64, complex128, complex256**: Números complejos representados por dos elementos de tipo float de 32, 64 o 128 bits respectivamente.
- **bool**: Tipo booleano para almacenar valores True y False
- **object**: Tipo genérico para objetos en Python
- **string** : Tipo string con longitud prefijada.
- **unicode** : Igual que **string**, para unicode.

Numpy: librería de cálculo numérico y álgebra lineal

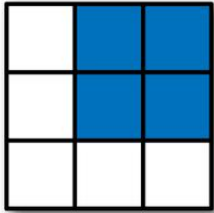
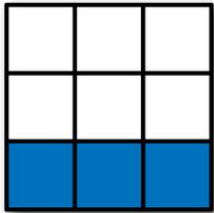
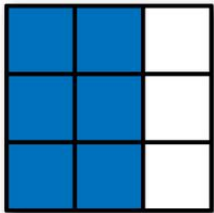
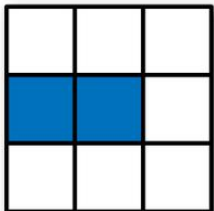


Numpy: librería de cálculo numérico y álgebra lineal

Algunos ejemplos de operaciones inteligentes de arrays



Numpy: librería de cálculo numérico y álgebra lineal

	Expression	Shape
	<code>arr[:2, 1:]</code>	<code>(2, 2)</code>
	<code>arr[2]</code>	<code>(3,)</code>
	<code>arr[2, :]</code>	<code>(3,)</code>
	<code>arr[2:, :]</code>	<code>(1, 3)</code>
	<code>arr[:, :2]</code>	<code>(3, 2)</code>
	<code>arr[1, :2]</code>	<code>(2,)</code>
	<code>arr[1:2, :2]</code>	<code>(1, 2)</code>

Subindexación en arrays

Numpy: números aleatorios y álgebra lineal

Otra de las componentes esenciales de **numpy** son los módulos:

- **numpy.random**- librería de generación de números aleatorios. Es fundamental para tareas como simulación y experimentación estadística.
- **numpy.linalg** - potente librería de álgebra lineal que emula a Matlab

Numpy: librería de cálculo numérico y álgebra lineal

Resumen

- **NumPy** es una librería a bajo nivel para tratar con matrices.
- Sirve como base y sustento en las tareas de **Pandas**.
- Proporciona muchos métodos y opciones para manejar datos eficientemente.
- Se pueden destacar de sus rasgos la facilidad para indexar elementos, definir tipos de datos y aplicar operaciones sobre todos los elementos de un array.
- Además de **Pandas**, **NumPy** es la base de muchas otras librerías enfocadas en el análisis y el modelado de datos como pueden ser **Scikit-learn** y **Tensorflow** .

Definiendo funciones en Python

```
def factorial(n): #se declara el nombre de la función y sus argumentos  
    c=np.prod(np.arange(1,n+1)) #en formato indentado se realizan las operaciones  
    return c #el valor a devolver  
  
#esto es un ejemplo con bucles dentro  
def fibonacci(n):  
    if n==1 or n==2:  
        c=1  
    else:  
        a0=1  
        a1=1  
        for k in range(n-1):  
            aux=a0+a1  
            a0=a1  
            a1=aux  
        c=a1  
  
    return c
```

```
factorial(6)
```

```
720
```

```
fibonacci(5)
```

```
8
```

Definiendo funciones paramétricas usando clases

- En programación científica y en machine learning es habitualmente necesario definir objetos que dependen de parámetros ó que tienen varios métodos aplicables sobre ciertas variables.
- Este tipo de objetos se traducen en forma de **clase** en Python. Se pueden entender como una extensión de las funciones con más posibilidades.
- La programación orientada a objetos es predominante en librerías de Machine Learning en Python como **scikit-learn**, que está enteramente estructurada de este modo.

```
class exponencial():  
  
    #esta es la partícula inicial que almacena los parámetros necesarios en la clase  
    def __init__(self,lamb):  
        self.lamb=lamb  
  
    #definimos los métodos, se puede llamar a los parámetros almacenados en __init__  
    def dens(self,x):  
        import numpy as np  
        lamb=self.lamb  
        return lamb*np.exp(-lamb*x)  
    def dist(self,x):  
        import numpy as np  
        lamb=self.lamb  
        return 1-np.exp(-lamb*x)
```

```
fexp=exponencial(2) #define una instancia de la clase con el parámetro lambda=3  
fexp.dens(4),fexp.dist(1) #aplica ambos métodos sobre el valor dado
```

```
(0.00067092525580502371, 0.8646647167633873)
```